

Optimal Alphabetic Ternary Trees

J. David Morgenthaler*

T. C. Hu†

February 14, 2014

Abstract

We give a new algorithm to construct optimal alphabetic ternary trees, where every internal node has at most three children. This algorithm generalizes the classic Hu-Tucker algorithm, though the overall computational complexity has yet to be determined.

1 Introduction

The optimal ternary alphabetic tree problem generalizes the well-studied binary case to consider trees with internal nodes having at most three children. This problem was mentioned in algorithm textbooks [3, 5], but has remained unsolved for 40 years..

First, let us briefly review the optimal binary alphabetic tree problem. Given a sequence of square nodes with weights $\{w_1, w_2, \dots, w_n\}$. Find the binary alphabetic tree with the least total path length. The problem was solved by the Hu-Tucker algorithm (see [3, 4]). There are two kinds of nodes, where every circular node has two children, and every square node has no children. A pair of nodes is called compatible if there is no square node between the pair. The weight of the parent node is the sum of weights of its children.

The Hu-Tucker algorithm has three phases:

Phase I: Combination

Start with a sequence of n square nodes. Keep combining the compatible pair of nodes with the least weight, creating a new circular node. In case of a tie, select the pair with the leftmost position. Keep combining until only one circular node (the root) remains.

Phase II: Assignment

Mark the level of each square node with its distance from the root in Phase I. Thus we obtain a sequence of level numbers $\{l_1, l_2, \dots, l_n\}$ corresponding to the n square nodes in Phase I.

*Google, Mountain View, CA 94303. Email: jdm@google.com.

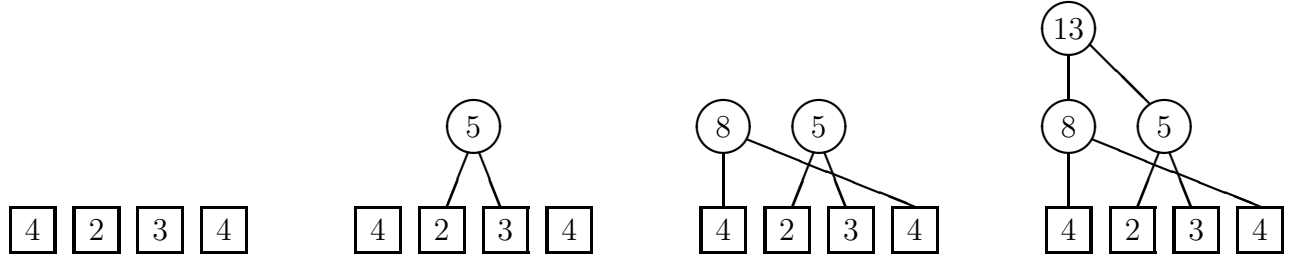
†Department of Computer Science and Engineering, University of California, San Diego, CA 92093. Email: hu@cs.ucsd.edu.

Phase III: Reconstruction

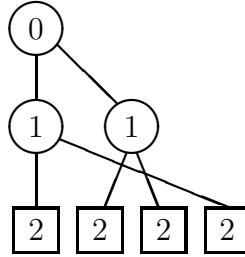
Start with the original sequence of n square nodes, and the corresponding level sequence. Find the adjacent pair of square nodes whose level numbers, say q , are maximum among all adjacent pairs. In the case of a tie, select the leftmost pair. Replace the adjacent pair in the sequence with a new parent node with level $q - 1$. Continue until the sequence consists of a single node with level 0. The resulting tree is the optimal binary alphabetic tree.

If we apply the Hu-Tucker Algorithm to the sequence of four square nodes with weights $\{4, 2, 3, 4\}$, then we have the following results.

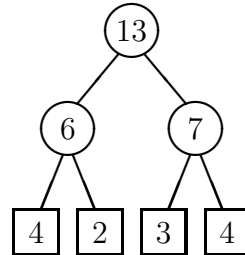
Phase I: Combination



Phase II: Assignment



Phase III: Reconstruction



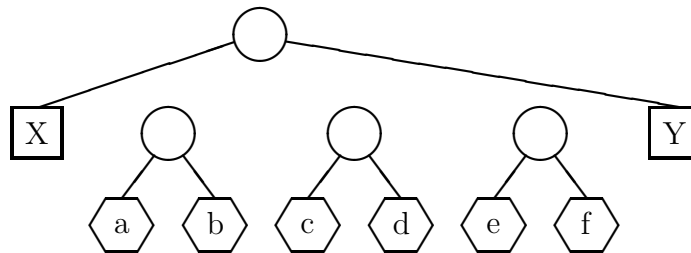
Definition 1 The combination of two compatible nodes with one or more circular nodes between them is called a cross-over combination. For example, the creation of a circular node with weight 8 by the combination of two square nodes each with weight 4 is called a cross-over.

Definition 2 A forest without any cross-over combinations is called a k -sum forest if there are k combinations. For example, in the Phase III reconstruction above, the resulting tree is a 3-sum forest.

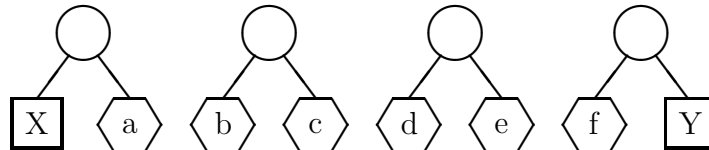
To prove correctness, we need two things:

1. We can always convert any tree with cross-overs to a forest with the same number of sums without and cross-over.
2. The final binary alphabetic tree has the minimum cost.

Proof of (1). Let us assume that our first cross-over has three circular nodes between the pair $X Y$



where hexagons denote either a circular or a square node. Then we can always do the four combinations as shown below.



Where in the last two figures, the total path lengths are the same, and both are created with 4 sums, and with the same alphabetic order. This proves that the final tree is alphabetic.

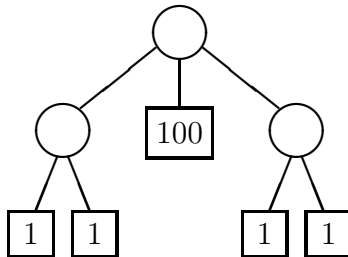
To prove that the final binary tree is optimal, *i.e.* of minimal cost, we can use induction on the sums during the algorithm. After the first k combinations, we can apply phases II and III to obtain an optimal k -sum forest. This leads to an interpretation of the weight of each circular node as the incremental cost of moving from an optimal $(k - 1)$ -sum forest to an optimal k -sum forest.

2 Optimal Ternary Alphabetic Trees

Given a sequence of square nodes with weights $\{w_1, w_2, \dots, w_n\}$ where every circular node has t children ($t \leq 3$), and every square node has no children. The optimal alphabetic ternary tree has the least total path length for the given sequence of weights.

This problem was mentioned by Knuth [5], p. 451, problem 40. He presented a simple five node sequence $\{1, 1, 100, 1, 1\}$ demonstrating the difficulty in applying a straight-forward

generalization of the Hu-Tucker algorithm to triples. The sequence shows that optimal trees may include interior nodes with two as well as three children, as seen in its optimal ternary tree below.



Knuth's example requires us to reconsider the idea of a *Permanent Circular Node* [2] from the binary case and additionally explore the impact of odd or even cardinality of a sub-sequence on the combination phase. To begin with, clearly a two square node initial sequence requires a binary circular node.

2.1 Binary Circular Nodes

Let our initial sequence be a set of square nodes with monotonically increasing weights $\{a, b, \dots\}$. If we have an odd number of nodes, our optimal ternary tree can consist of internal nodes with exactly three children each. It is easy to see that a tree that instead contains any binary nodes must cost more.

On the other hand, if the number of nodes in the initial sequence is even, we must have at least one binary combination. The following example shows that we should do that binary combination in the beginning. For four square nodes of weights a, b, c, d , where $a < b < c < d$



On the left, the total cost $= 2a + 2b + c + d$, while on the right, the total cost $= 2a + 2b + 2c + d$.

If the optimal tree contains binary nodes, they are at the bottom combining two squares. Specifically, they will be minimum adjacent pair(s). Knuth's sequence has two such pairs, each meeting the *Permanent Circular Node* criteria we described for optimal binary alphabetic trees [2]. We generalize this idea to the ternary case.

2.2 Generalized Permanent Circular Nodes

Consider a subsequence of adjacent square nodes with weights w_i, \dots, w_j where their combined weight is smaller than either neighbor w_{i-1} or w_{j+1}

$$w_{i-1} > w_i + \dots + w_j < w_{j+1}$$

For convenience, we can add nodes w_0 and w_{n+1} of infinite weight at either end of the initial sequence to handle edge cases. These added nodes never combine, but are simply an algorithmic convenience.

Generalizing Gilbert and Moore's Theorem 5 [1] to the ternary case is straightforward, but results in two possibilities:

1. All nodes in the subsequence combine to form a single circle before combination with either larger adjacent node.
2. Two nodes, rather than a single one, are available to combine with one neighbor.

The decision as to which of the two possibilities to use will be determined at the next level, by the evenness or oddness of the cardinality of that larger subproblem. Taken to the top level, the complete sequence can also be viewed as a 'permanent' circular node between the two added infinite-weight nodes. For this final case, of course, we ignore the two node possibility.

Applying this idea to Knuth's example, the five node sequence $\{1, 1, 100, 1, 1\}$ contains only three nodes considering the generalized permanent nodes as units: $\{(1, 1), 100, (1, 1)\}$. Three is odd, and so forms a ternary circle, implying that the two permanent nodes each combine to form a single circle. We can increase the number of squares in each permanent node to see the idea in operation.

Adding a third square to the initial permanent node, $\{1, 1, 1, 100, 1, 1\}$ results in a similar situation as before, just turning the first pair into a triple: $\{(1, 1, 1), 100, (1, 1)\}$. If we add a fourth square to that group, only those four nodes are affected, and the four node sub-problem will provide a single permanent circular to the next level as before, consisting of one pair as we saw in the previous section.

Now consider instead adding another heavy node to Knuth's example sequence:

$$\{1, 1, 100, 100, 1, 1\}$$

. The cardinality of the larger problem is even, and one of the permanent circular nodes must provide two nodes, rather than just a single one: $\{(1), (1), 100, 100, (1, 1)\}$. The resulting five node sequence has optimal ternary tree $\{((1, 1, 100), 100, (1, 1))\}$, where each internal node with children a, b appears between parentheses (a, b) .

2.3 Pure Ternary Case

As a simplifying constraint, we now consider the case where $t = 3$. That is, every circle must have exactly three children. For this case, the initial sequence has an odd number of nodes,

and we avoid any hierarchal sub-problems by disallowing permanent circular nodes. That is, for every adjacent pair w_i, w_{i+1} , either

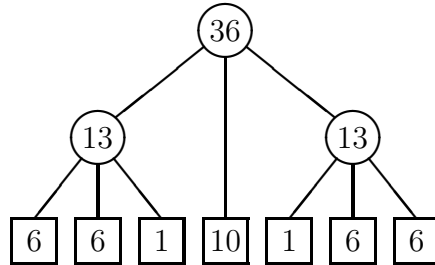
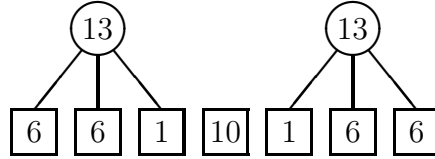
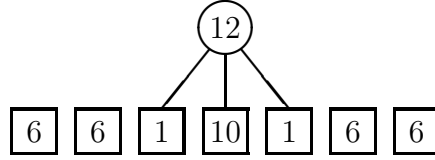
$$w_{i-1} \leq w_i + w_{i+1}$$

or

$$w_i + w_{i+1} \geq w_{i+2}$$

For this purpose only, we consider added negative infinite weight nodes at the ends of the sequence.

This problem was explored by by Hu and Shing [3], pp. 196-7. Their counter-example to a naive application of the Hu-Tucker algorithm uses the initial weight sequence $\{6, 6, 1, 10, 1, 6, 6\}$. Approaching this problem from the viewpoint of consecutive optimal k -sum forests, for $k \in \{1, 2, 3\}$:



Note that the associated level numbers of the seven square nodes, at each step, are

$$\begin{array}{ccccccc} 0, & 0, & 1, & 1, & 1, & 0, & 0 \\ 1, & 1, & 1, & 0, & 1, & 1, & 1 \\ 2, & 2, & 2, & 1, & 2, & 2, & 2 \end{array}$$

where the central square node with weight 10 first has level 1, which then decreases to zero, and finally increases back to level 1 again. This differs from the binary case where levels

increase monotonically as more combinations are made. Any Hu-Tucker type ternary algorithm must allow for such level reduction during the combination phase, since the combination phase can always be stopped after k combinations, yielding the level forest for the optimal k -sum alphabetic forest.

3 Ternary Algorithm

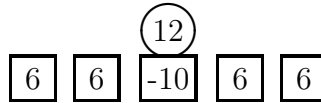
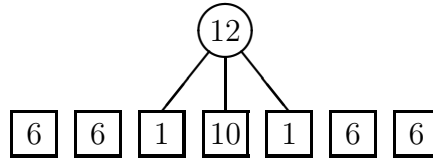
Our new algorithm uses the same three phases as the Hu-Tucker binary algorithm. Only the combination phase operates differently, as we must generalize the idea of compatible pair to compatible triple and define a new concept, the accordion. A set of three nodes is called a compatible triple if there is no square nodes between the three nodes.

3.1 Accordions

The main new concept in our algorithm is the accordion. An accordion forms the central node of a combining triple, and consists of alternating positive and negative weight nodes. The outer nodes of an accordion always have positive weight, with the smallest accordion consisting of a single node. In the example of sequence $\{6, 6, 1, 10, 1, 6, 6\}$, the first accordion is the central square node with weight 10.

The next combination in the algorithm involves a negative weight, the previously combined central node of weight 10, this time with weight -10 . The accordion consists of the three central nodes, with weight $(+6, -10, +6) = 2$, giving the new circle the weight $6 + 2 + 6 = 14$.

Negative weight nodes are previously combined square nodes at level 1. After this combination, any negative weight nodes reappear in the sequence as square nodes, in their original position. Continuing the example, the sequence after each phase I combination appears below.



The final combination creates a circle of weight 36 with the three remaining nodes as children. Note that the square with weight 10 has participated in each combination, twice with positive weight, and once with negative weight.

A square child of a circle in the current sequence is available with a negative weight if that node is the central child of a top-level internal node in the unique optimal k -sum forest obtained from the current sequence. Reappearing squares are positioned below the circle with which they were associated. Once a level 1 node associated with a particular circle has appeared as negative weight node in a combination, it cannot be used again. However, the reappearing square node may later combine into a different circle, and again be used as a negative component of a later accordion. Once its circular parent combines, a square loses its availability. Thus, only squares at level 2 need be considered.

During phase II, each occurrence of a negative weight in the phase I tree contributes a negative level number, which when combined with the positive occurrences of the same square yields the overall level for that square in the phase III reconstruction.

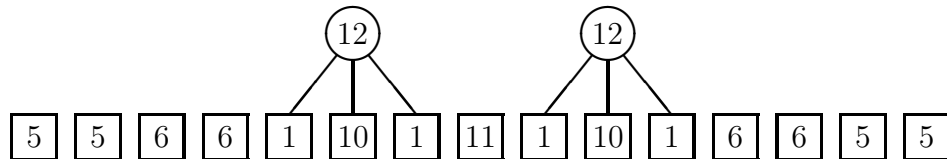
The proof of correctness and optimality of this algorithm is similar to the binary case, by induction on the k -sum forest created by ending phase I after the first k combinations and proceeding with phases II and III.

4 General Case

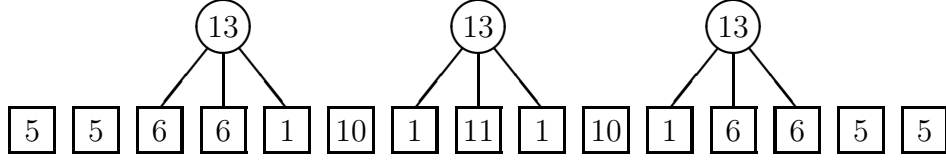
Putting the two ideas together yields the general optimal ternary tree algorithm. First, identify permanent circular nodes with a linear scan of the input. Each such node is a subproblem with two solutions - an optimal ternary tree with a single root node, or an optimal forest with two roots. Permanent circular nodes may be hierarchical, and the procedure is repeated up the hierarchy until all remaining squares are almost uniform, that is no remaining pairs have smaller combined weight than both their adjacent neighbors.

5 Example

Consider initial sequence $\{5, 5, 6, 6, 1, 10, 1, 11, 1, 10, 1, 6, 6, 5, 5\}$. After the first two combinations, we have a straight forward 2-sum forest with total weight 24:



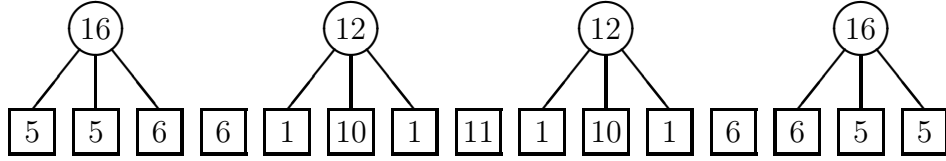
For the next Phase I combination, we find a large accordion $(+6, -10, +11, -10, +6) = 3$, which participates in minimum triple $(6, (+6, -10, +11, -10, +6), 6)$ of weight 15, yielding optimal 3-sum forest of total weight $24 + 15 = 39$:



An even larger accordion forms the central node of the forth triple combination,

$$(5, (+5, -6, +10, -11, +10, -6, +5), 5)$$

of weight 17, yielding an optimal 4-sum forest with total weight $39 + 17 = 56$:



At this point, the fifth combination is the three square triple $(6, 11, 6)$, for an optimal 5-sum forest with all nodes at level 1, with weight $56 + 23 = 79$. Now our sequence consists of only circular nodes, which we place in a queue: $(12, 12, 15, 17, 23)$. Two more combinations - $12+12+15 = 39$ and $17+23+39 = 79$ completes Phase I giving a total cost of $79+39+79 = 197$.

For each combination, we can also show the current level of each square using underscores and overscores. We denote a combination of positive weight nodes with an underscore, while reappearing squares of negative weight receive an overscore. Each underscore represents an increase in level, while each overscore is a decrease. Using this scheme, the Phase I combinations above look like this:

$$\begin{array}{cccccccccccccccc}
 5 & 5 & 6 & 6 & \underline{1} & \underline{10} & \underline{1} & 11 & \underline{1} & \underline{10} & \underline{1} & 6 & 6 & 5 & 5 \\
 5 & 5 & \underline{6} & \underline{6} & \underline{1} & \overline{10} & \underline{1} & 11 & \underline{1} & \overline{10} & \underline{1} & \underline{6} & \underline{6} & 5 & 5 \\
 5 & 5 & 6 & \overline{6} & \underline{1} & \overline{10} & \underline{1} & \overline{11} & \underline{1} & \overline{10} & \underline{1} & \overline{6} & 6 & 5 & 5 \\
 5 & 5 & 6 & \overline{6} & \underline{1} & \overline{10} & \underline{1} & \overline{11} & \underline{1} & \overline{10} & \underline{1} & \overline{6} & 6 & 5 & 5 \\
 5 & 5 & \underline{6} & \overline{6} & \underline{1} & \overline{\overline{10}} & \underline{1} & \overline{11} & \underline{1} & \overline{\overline{10}} & \underline{1} & \overline{6} & 6 & 5 & 5 \\
 5 & 5 & \underline{6} & \overline{\overline{6}} & \underline{1} & \overline{\overline{10}} & \underline{1} & \overline{11} & \underline{1} & \overline{\overline{10}} & \underline{1} & \overline{\overline{6}} & 6 & 5 & 5
 \end{array}$$

Phase II thus yields level numbers:

$$2, 2, 3, 3, 3, 2, 3, 3, 3, 2, 3, 3, 3, 2, 2$$

6 Conclusion and Future Work

We have extended the Hu-Tucker algorithm to the ternary case, where each internal node has at most three children. The ternary case is more complex than the binary alphabetic tree problem, and future work includes determination of the overall complexity of the solution presented here. There is additional work of tracking two possibilities for each permanent circular node and computing the oddness or evenness of each subproblem. Accordians must also be tracked, and their existence adds work to determining the next phase I combination.

The question of whether a Hu-Tucker type level tree algorithm can be further extended to handle $t > 3$ children remains open.

References

- [1] E. N. Gilbert and E. F. Moore. Variable length binary encodings. *Bell System Technical Journal*, 38:933–968, 1959.
- [2] T. C. Hu and J. D. Morgenthaler. Optimum alphabetic binary trees. In *Combinatorics and Computer Science: 8th Franco-Japanese and 4th Franco-Chinese Conference*, Lecture Notes in Computer Science, volume 1120, pages 234–243. Springer-Verlag, 1996.
- [3] T. C. Hu and M. T. Shing. *Combinatorial Algorithms, Second Edition*. Dover, 2002.
- [4] T. C. Hu and A. C. Tucker. Optimal computer search trees and variable-length alphabetic codes. *SIAM Journal on Applied Mathematics*, 21(4):514–532, 1971.
- [5] D. E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching*. Addison-Wesley, 1973.